

1 CLAIMS

2

3 1. A method used in managing a serverless distributed file system, the

4 method comprising:

5 managing directories of the file system using Byzantine groups; and

6 managing files within the directories without using Byzantine groups.

7

8 2. A method as recited in claim 1, further comprising managing files

9 within the directories by saving replicas of the files to fewer computers than exist

10 in the Byzantine groups.

11

12 3. A method as recited in claim 1, wherein each directory includes one

13 or more directory entries corresponding to one or more of the files, and wherein

14 each directory entry includes:

15 an identification of the file;

16 an identification of a plurality of computers where replicas of the file are

17 stored; and

18 file verification data.

19

20 4. A method as recited in claim 3, wherein the file verification data

21 comprises a hash value generated from applying a cryptographically secure hash

22 function to the file.

23

24

25

1           5.    A method as recited in claim 3, wherein the file verification data  
2 comprises a file identification number, a file version number, and a name of a user  
3 whose signature is on the file.  
4

5           6.    A method as recited in claim 1, wherein the directories are managed  
6 using a hierarchical namespace.  
7

8           7.    A method as recited in claim 1, wherein each of a plurality of  
9 computers in the serverless distributed file system need not trust the other ones of  
10 the plurality of computers.  
11

12           8.    A serverless distributed file system comprising:  
13           a plurality of computers;  
14           a first set of the plurality of computers operating to store directory  
15 information for the file system, wherein each computer of the first set is part of a  
16 directory Byzantine group; and  
17           a second set of the plurality of computers operating to store replicas of the  
18 files in the file system, wherein for each file stored in the file system a plurality of  
19 replicas of the file are stored on the second set of computers, and wherein the  
20 quantity of replicas is less than the quantity of computers in the Byzantine group.  
21

22           9.    A serverless distributed file system as recited in claim 8, wherein one  
23 or more of the plurality of computers are in both the first set and the second set.  
24  
25

1           **10.**    A serverless distributed file system as recited in claim 8, wherein the  
2 directory information includes a plurality of directory entries, and wherein each  
3 directory entry on a computer in the first set includes an indication of each  
4 computer in the second set where a copy of a file corresponding to the directory  
5 entry is located.

6  
7           **11.**    A serverless distributed file system as recited in claim 8, wherein the  
8 replicas comprise encrypted versions of the file being stored.

9  
10          **12.**    A method comprising:  
11           generating a directory entry corresponding to a file to be stored in a  
12 serverless distributed file system;  
13           saving the directory entry to each of a first plurality of computers that are  
14 part of a Byzantine-fault-tolerant group; and  
15           saving the file to each of a second plurality of computers, wherein fewer  
16 computers are in the second plurality of computers than are in the first plurality of  
17 computers, and wherein at least one of the second plurality of computers is not  
18 part of the Byzantine-fault-tolerant group.

19  
20          **13.**    A method as recited in claim 12, further comprising:  
21           receiving, prior to saving the file to each of the second plurality of  
22 computers, the file in encrypted form.

1           **14.**    A method as recited in claim 12, further comprising:  
2           receiving, prior to generating the directory entry, an encrypted file name of  
3 the file.

4  
5           **15.**    A method as recited in claim 12, wherein at least one of the second  
6 plurality of computers is part of the Byzantine-fault-tolerant group.

7  
8           **16.**    A computer comprising:  
9           a processor;  
10          a memory coupled to the processor; and  
11          wherein the memory is to store a plurality of instructions to implement a  
12 file system using a hierarchical namespace to store files, wherein the file system is  
13 distributed across a plurality of computers including the computer, wherein each  
14 of the plurality of computers can operate as both a client computer and a server  
15 computer, wherein each of the plurality of computers need not trust the other ones  
16 of the plurality of computers, wherein files and corresponding directory entries are  
17 stored in the file system, and wherein for any given file fewer copies of the file are  
18 stored than are copies of the corresponding directory entry.

19  
20          **17.**    A computer as recited in claim 16, wherein the file system stores  
21 directory entries using Byzantine groups and objects corresponding to the  
22 directory entries without using Byzantine groups.

1       **18.**     A computer as recited in claim 16, wherein the computer is part of a  
2 directory group responsible for managing a set of directories in the hierarchical  
3 namespace, and wherein the plurality of instructions are further to cause the  
4 computer to identify a new directory group and delegate responsibility for  
5 managing a subset of the set of directories to the new directory group.

6  
7       **19.**     A computer as recited in claim 18, wherein the plurality of  
8 instructions are further to cause the computer to:

9       identify a group of computers to be part of the new directory group;

10      generate a delegation certificate for the subset;

11      digitally sign the delegation certificate; and

12      issue the delegation certificate to the group of computers.

13  
14      **20.**     A computer as recited in claim 16, wherein the computer includes a  
15 cache of pathname to directory group mappings, and wherein the plurality of  
16 instructions are further to cause the computer to determine where to locate a file  
17 corresponding to a pathname in the file system by performing the following acts:

18       checking the cache to determine a mapping for a longest prefix of a desired  
19 pathname; and

20       if the entire pathname is mapped to a directory group using the cache, then  
21 accessing a member of the directory group to determine where to locate the file,  
22 and otherwise repeating the following until the entire pathname is mapped to a  
23 directory group,

1 obtaining, from a member of the directory group corresponding to  
2 the longest prefix of the desired pathname, mappings for a relevant subtree  
3 from the longest prefix, and

4 if the entire pathname is not mapped to a directory group using the  
5 relevant subtree from the longest prefix, then repeating the obtaining with  
6 the longest prefix being the previously used longest prefix concatenated  
7 with the relevant subtree.

8  
9 **21.** A method comprising:

10 identifying a group of computers to which a subtree of a hierarchical  
11 namespace used to store files is to be delegated;

12 generating a delegation certificate for the subtree;

13 digitally signing the delegation certificate; and

14 issuing the delegation certificate to the group of computers.

15  
16 **22.** A method as recited in claim 21, wherein digitally signing the  
17 delegation certificate comprises having the delegation certificate digitally signed  
18 by a plurality of computers.

19  
20 **23.** A method as recited in claim 21, wherein the group of computer  
21 comprise a Byzantine-fault-tolerant group.

1           **24.**    A method as recited in claim 21, wherein the delegation certificate  
2 comprises:

3           a first digitally signed certificate identifying another group of computers  
4 responsible for managing a namespace root of the subtree; and

5           a second digitally signed certificate allowing authorization of the group of  
6 computers to manage the subtree to be traced to the other group of computers  
7 responsible for managing the namespace root.

8  
9           **25.**    A method as recited in 24, wherein the second digitally signed  
10 certificate comprises:

11           an identification of a path below the beginning of another subtree  
12 previously delegated to a third group of computers, wherein the third group of  
13 computers are the directory group performing generating;

14           an identification of a root of the other subtree delegated to the third group  
15 of computers;

16           an identification of the subtree; and

17           an identification of the members of the group of computers.

18  
19           **26.**    A method as recited in claim 25, wherein the computers in the third  
20 group of computers are the same computers as in the other group of computers.

21  
22           **27.**    A method as recited in 24, wherein the first digitally signed  
23 certificate is digitally signed by a certification authority (CA).

1           28.     A method as recited in 24, wherein the delegation certificate further  
2 comprises one or more additional digitally signed certificates allowing a certificate  
3 chain to be established from the second digitally signed certificate to the first  
4 digitally signed certificate.

5  
6           29.     A method implemented in a computing device of a serverless  
7 distributed file system, the method comprising:

8           checking a local cache of pathname to Byzantine-fault-tolerant directory  
9 group mappings to determine a mapping for a longest prefix of a desired  
10 pathname; and

11           if the entire pathname is mapped to a Byzantine-fault-tolerant directory  
12 group using the local cache, then accessing a member of the Byzantine-fault-  
13 tolerant directory group to determine where to locate a file corresponding to the  
14 pathname, and otherwise repeating the following until the entire pathname is  
15 mapped to a Byzantine-fault-tolerant directory group,

16           obtaining, from a member of the Byzantine-fault-tolerant directory  
17 group corresponding to the longest prefix of the desired pathname,  
18 mappings for a relevant subtree from the longest prefix, and

19           if the entire pathname is not mapped to a Byzantine-fault-tolerant  
20 directory group using the relevant subtree from the longest prefix, then  
21 repeating the obtaining with the longest prefix being the previously used  
22 longest prefix concatenated with the relevant subtree.



1       **30.**     A method as recited in claim 29, wherein the mappings are obtained  
2 from the member of the directory group via a delegation certificate digitally signed  
3 by one or more members of the directory group.  
4

5       **31.**     A method as recited in claim 29, further comprising adding the  
6 mappings for the relevant subtree to the local cache.  
7

8       **32.**     A method as recited in claim 29, wherein the longest prefix includes  
9 one or more directories in addition to a namespace root.  
10

11       **33.**     A method implemented in a serverless distributed file system, the  
12 method comprising:  
13

14       receiving a request to open an object with one or more selected locks;

15       checking whether the one or more selected locks conflict with a lock  
16 already granted to another application, wherein at least one of the selected locks  
17 represents the right to open the object without sharing an open mode; and  
18

19       granting the request to open the object only if the one or more selected  
20 locks do not conflict with a lock already granted to another application.  
21

22       **34.**     A method as recited in claim 33, further comprising:  
23

24       requesting that the device that holds the lock that conflicts with the one or  
25 more selected locks return the conflicting lock; and  
26

27       granting the request to open the object if the conflicting lock is returned,  
28 otherwise denying the request to open the object.  
29

1       **35.**     A method as recited in claim 33, further comprising:  
2       upgrading the request to a broader scope than indicated in the request;  
3       checking whether the one or more selected locks of the broader scope  
4       conflict with a lock already granted to another application; and  
5       granting the request to open the object with the broader scope only if the  
6       checking of the one or more selected locks of the broader scope indicate that no  
7       conflict exists.

8  
9       **36.**     A method as recited in claim 33, further comprising denying the  
10      request if checking whether the one or more selected locks indicate a desire to  
11      share the object for one or more operations that conflict with sharing indicated by  
12      another application that has previously been granted a lock or checking whether  
13      the one or more selected locks conflict with a lock already granted to another  
14      application indicate a conflict exists.

15  
16      **37.**     A method as recited in claim 33, wherein granting the request  
17      further comprises upgrading the request to a broader scope than indicated in the  
18      request and granting the broader scope.

19  
20      **38.**     A method as recited in claim 33, further comprising attempting to  
21      downgrade the previous lock to a narrower scope than previously granted to  
22      another application prior to denying the request if the one or more selected locks  
23      conflict with a lock already granted to another application.  
24  
25

1           **39.**    A method as recited in claim 33, wherein the object comprises a file.

2  
3           **40.**    A method as recited in claim 33, wherein the object comprises a  
4 directory.

5  
6           **41.**    A method as recited in claim 33, wherein the one or more selected  
7 locks comprise an Open Read lock.

8  
9           **42.**    A method as recited in claim 33, wherein the one or more selected  
10 locks comprise an Open Write lock.

11  
12           **43.**    A method as recited in claim 33, wherein the one or more selected  
13 locks comprise an Open Delete lock.

14  
15           **44.**    A method as recited in claim 33, wherein the at least one selected  
16 lock comprises a Not Shared Read lock.

17  
18           **45.**    A method as recited in claim 33, wherein the at least one selected  
19 lock comprises a Not Shared Write lock.

20  
21           **46.**    A method as recited in claim 33, wherein the at least one selected  
22 lock comprises a Not Shared Delete lock.

1       **47.** One or more computer readable media having stored thereon a  
2 plurality of instructions to implement a serverless distributed file system, wherein  
3 the plurality of instructions, when executed by one or more processors, causes the  
4 one or more processors to perform acts comprising:

5       assigning responsibility for managing one or more directories to a directory  
6 group, wherein each member of the directory group is a computer participating in  
7 the serverless distributed file system; and

8       employing a plurality of locks to control access to objects in each directory,  
9 wherein the plurality of locks comprise,

10       a first set of locks to control opening of the objects, and

11       a second set of locks to control access to the data in the objects.

12  
13       **48.** One or more computer readable media as recited in claim 47,  
14 wherein the second set of locks comprises:

15       a Read lock to control read access to the data in the objects; and

16       a Write lock to control write access to the data in the objects.

17  
18       **49.** One or more computer readable media as recited in claim 47,  
19 wherein the first set of locks comprises:

20       an Open Read lock to control opening of the objects for reading;

21       an Open Write lock to control opening of the objects for writing; and

22       an Open Delete lock to control opening of the objects for deleting.  
23  
24  
25

1           **50.**   One or more computer readable media as recited in claim 47,  
2 wherein the first set of locks comprises:

3           a Not Shared Read Lock to indicate an unwillingness to share the ability to  
4 read the objects.

5  
6           **51.**   One or more computer readable media as recited in claim 47,  
7 wherein the first set of locks comprises:

8           a Not Shared Write Lock to indicate an unwillingness to share the ability to  
9 write to objects.

10  
11           **52.**   One or more computer readable media as recited in claim 47,  
12 wherein the first set of locks comprises:

13           a Not Shared Delete Lock to indicate an unwillingness to share the ability  
14 to delete the objects.

15  
16           **53.**   One or more computer readable media as recited in claim 47,  
17 wherein the plurality of locks further comprises an Insert lock to control creation  
18 of a new object with a particular name.

19  
20           **54.**   One or more computer readable media as recited in claim 47,  
21 wherein the plurality of locks further comprises an Exclusive lock to obtain all of  
22 the other ones of the plurality of locks for an object.

1           **55.** One or more computer readable media as recited in claim 47,  
2 wherein the objects comprise one or more files and one or more directories.

3  
4           **56.** A serverless distributed file system comprising:  
5 a plurality of computers;  
6 a first set of the plurality of computers operating to store directory  
7 information for the file system, wherein each computer of the first set is part of a  
8 Byzantine-fault-tolerant group;

9 a second set of the plurality of computers operating to store replicas of the  
10 files in the file system, wherein for each file stored in the file system a plurality of  
11 replicas of the file are stored on the second set of computers, and wherein fewer  
12 computers are in the first set than in the second set;

13 wherein the first set of computers is configured to delegate management  
14 responsibility for a group of directories of the file system to a third set of the  
15 plurality of computers by,

16 generating a delegation certificate for the group of directories,  
17 digitally signing the delegation certificate, and

18 issuing the delegation certificate to the third set of computers; and

19 wherein the third set of computers is configured to maintain management  
20 responsibility for the group of directories by employing a plurality of locks to  
21 control access to objects in each directory of the group, wherein the plurality of  
22 locks include,

23 a first set of locks to control opening of the objects, and

24 a second set of locks to control access to the data in the objects.  
25